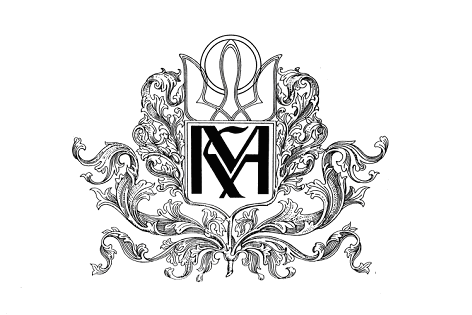
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мережних технологій



**КУРСОВА РОБОТА**

**за спеціальністю „Програмна інженерія” 6.050103**

**РОЗРОБКА СЕРВІСНИХ УТИЛІТ ЗАСОБАМИ**

**КОМАНДНИХ ФАЙЛІВ WINDOWS**

Керівник курсової роботи

к.т.н., доц. Франчук О. В.

\_\_\_\_\_\_\_\_\_\_\_

(підпис)

“\_\_\_\_” \_\_\_\_\_\_\_\_\_ 2015 р.

Виконав

студент ІІ курсу

факультету інформатики

Савченко С. О.

КИЇВ — 2015

**ЗМІСТ**

**Вступ**……………………………………………………………………………4

1. **Огляд командних файлів Windows**……………………………………...6

1.1 Що таке командний файл?.......................................................................6

1.2 Створення командного файлу………………………………………….6

1.3 Структура командного файлу………………………………………….7

1.4 Основні команди………………………………………………………..8

1.4.1 Робота з файлами……………………………………….………..8

1.4.2 Атрибути файлів……………………………………………...….9

1.4.3 Вікно виконання CMD…………………………...……………...9

1.4.4 Паузи між командами………………………...….……………..10

1.4.5 Системні налаштування……………………………...………...10

1.4.6 Команда CALL…………………………………………….........11

1.4.7 Перенаправлення вводу/виводу даних…………………...…...12

1.5 Ключі запуску………….………………………………………………13

1.6 Змінні…………………………………………………………………...14

1.7 Конструкція GOTO……………………………………………………16

1.8 Оператори умови IF…………………………………………………...17

1.9 Цикл FOR………………………………………………………………18

1.10 Обробка коду завершення програми (errorlevel)……………….......19

**2. Windows Script Host (WSH)**………………………………………………21

2.1 Огляд WSH ………………………………….………………………....21

2.2 Практичне використання………………………….…………………..21

**3. Застосування командних файлів Windows**…………………………….23

3.1 Для чого використовують командні файли Windows……………….23

3.2 Утиліти Windows NT Resource Kit……………………..……………..24

3.3 Приклади застосування командних файлів………………………….26

3.3.1 Утиліта для пошуку файлів ……………………………….…...26

3.3.2 Утиліта для пошуку папок ………………………………....….27

3.3.3 Утиліта для пошуку файлів за форматом ………………...…..28

3.3.4 Створення нового файлу ……………………………………...28

3.3.5 Автоматизований вхід на електронну пошту………...………29

3.3.6 Менеджер задач ………………………………………………..30

3.3.7 Видалення тимчасових файлів………………………………...31

3.3.8 Автоматичне створення резервних копій …….………………31

3.3.9 Перевірка відповідей………….………………………………..32

**Висновки**…………………………………………………………………….34

**Список використаної літератури**………………………………………...35

**Вступ**

В операційній системі Windows команди виконуються за допомогою так званого командного інтерпретатора або командного процесора. Командний (або пакетний) файл – це файл, в якому записані команди операційної системи, а також деякі інші додаткові інструкції. Підтримка командних файлів з’явилася в операційних системах MS-DOS.

Основним призначенням пакетного файлу є автоматизація послідовного виконання команд, що дає певну перевагу над ручним виконанням команд через командну стрічку завдяки деяким вбудованим конструкціям, таких як оператори if, goto та for, можливості яких були значно розширені в сімействі операційних систем Windows NT.

Командні файли широко застосовуються для автоматизації роботи користувача в операційній системі. В командних файлах використовуються команди конкретної операційної системи та її сервісного оточення. Крім того, є ряд команд, спеціально створених для управління роботою командного файлу.

Пакетні файли корисні для автоматичного запуску додатків. Основна область застосування - автоматизація найбільш рутинних операцій, які регулярно доводиться здійснювати користувачеві комп'ютера: наприклад, копіювання, переміщення, перейменування, видалення файлів; робота з папками, архівація та інше.

**Мета і завдання дослідження.** Мета цієї роботи визначити особливості створення та використання командних файлів в операційній системі Windows, проаналізувати функціонал, який забезпечується, експериментально дослідити та описати прикладні задачі, виконання яких можна полегшити за допомогою командних файлів.

**Об’єкт дослідження.** Командні файли Windows та скриптова мова batch language, створення їх засобами сервісних утиліт.

**Методи дослідження.** Використання різноманітної літератури, а також практичні дослідження під час самостійного створення командних файлів.

**Джерела дослідження.** Друковані видання, статті та інтернет ресурси.

**Практичне значення одержаних результатів.** В даній роботі Ви знайдете велику кількість корисних командних файлів, призначених для автоматизації рутинних задач під час роботи з операційною системою Windows.

**1. Огляд командних файлів Windows**

1.1 Що таке командний файл?

Командний (або пакетний) файл - це тип скриптового файлу в операційних системах DOS, OS/2 та Windows. Він складається з послідовності команд, які виконує інтерпретатор командного рядка, і зберігається у вигляді звичайного текстового файлу. Командний файл може містити будь-яку команду операційної системи. Також реалізована можливість використання конструкцій, що містять умовний перехід та цикли.

Командні файли були створені, щоб полегшити роботу, необхідну для виконання певних рутинних завдань, пов’язаних, переважно, з роботою з файлами, дозволяючи кожному користувачеві створювати власні сценарії для автоматизації певних завдань. Після запуску командного файлу, програма-оболонка (англ. shell program) читає файл стрічка за стрічкою і виконує команди.

Командні файли мають розширення .bat або .cmd, яке було додано в операційних системах Windows NT. Пакетний файл Windows – аналог скриптових файлів командного рядка (shell script) в Unix-подібних операційних системах (з розширенням .sh або .bsh).

1.2 Створення командного файлу

Командний файл – це звичайний текстовий файл, отже щоб створити порожній текстовий файл можна спочатку створити новий текстовий файл, а потім задати йому розширення .bat або .cmd. Редагувати командний файл можна за допомогою вбудованого в операційну систему Windows текстового редактора.

Також можна використовувати більш професійні засоби, наприклад, вільний текстовий редактор Notepad++. Використання подібних засобів робить процес розробки більш комфортним завдяки таким функціям як підсвічування синтаксису, автодоповнення коду та можливості зберігати файли в різних системах кодування.

1.3 Структура та синтаксис командного файлу

Кожен командний файл складається з послідовно записаних команд, а також певних додаткових конструкцій. Для командних файлів не існує суворої стандартизації вигляду вмісту. Зазвичай тіло файлу починається з команди @echo off, що відключає вивід на екран самих команд, наприклад:

**echo Testing**

**Ця команда виведе на екран саму команду –“echo Testing”, а на наступний рядок –результат її виконання – слово “Testing”.**

**@echo Testing**

**В результаті виконання цієї програми буде виведено лише слово “Testing”.**

**@echo off**

Ця команда вимикає вивід на екран усіх команд на час виконання всього командного файлу.

Якщо після виконання усіх команд необхідно залишити вікно виконання відкритим, у кінець файлу можна додати команду pause, тоді для закриття вікна треба бути натиснути будь-яку клавішу.

Усі команди та оператори є регістронезалежними, тобто, команди ECHO, echo та Echo абсолютно ідентичні. Під час написання командного файлу можна вставляти порожні стрічки та проводити табуляцію рядків для кращого сприйняття коду.

Для оформлення коментарів існує дві команди – rem або ::. Приклад коментарів:

rem Цей текст не буде

:: виведено на екран

Для будь-якої команди можна задати певні параметри. Повну інформацію щодо атрибутів певного файлу можна отримати виконавши команду [command] /?, яка виведе на екран детальну довідку.

1.4 Основні команди

1.4.1 Робота з файлами

start шлях\файл.формат

start http://google.com

Ця команда виконую запуск вказаного файлу. Також є можливість відкривати посилання на Інтернет-сторінки, вказуючи замість шляху до файлу потрібне посилання, яке відкриється в браузері за замовченням.

cd шлях

Зміна каталогу

md шлях\ім’я папки

Команда створює папку в заданому каталозі.

rmdir шлях\ім’я папки

Видалення папки.

Del шлях\ім’я файлу

Видалення файлу. Також, якщо треба виконати одну й ту саму операцію над багатьма файлами, можна використовувати маски – опис файлу за його ім’ям або розширенням. Наприклад команда

Del \*.txt

видалить усі текстові файли в поточному каталозі.

Ren шлях\назва шлях\нова назва

Перейменування файлу або папки.

Move папка папка

або

Move файл папка

Переміщення папки або файлу до вказаного каталогу. Аналогічна команда Copy - копіювання папки або файлу.

Print текстовий файл

Друк текстового файлу.

More < текстовий файл

Ця команда виводить текст з текстового документу у вікно CMD.

1.4.2 Атрибути файлів

Атрибут файлу – це метадані, що описують файл. Атрибут може знаходитись у двох станах – встановлений, або знятий. В операційних системах DOS та Windows існують чотири основні атрибути:

* Архівний – означає, що файл було змінено з часу проведення останнього резервного копіювання.
* Прихований – означає, що за замовчанням файл не буде відображатися в Провіднику та інших програмах.
* Системний – файл вважається системним, тобто необхідним для нормального функціонування операційної системи.
* Тільки читання – вміст файлу не можна змінювати.

Приклад:

attrib –A –H –S +R 123.txt

для файлу 123.txt були зняти атрибути *архівний, прихований, системний*, та присвоєно атрибут файлу *тільки для читання*.

1.4.3 Вікно виконання CMD

За допомогою команди @title Можна змінити назву вікна:

@title Текст заголовку

Для коректного відображення кирилиці в консолі існує декілька шляхів. В текстовому редакторі (наприклад Notepad++) треба вибрати пункт меню Кодування => Кодування => Кирилиця => OEM-866 і зберегти зміни. Крім того, можна змінити кодування сторінки консолі, вставивши на початку командного файлу стрічку chcp 1251 >nul.

Можна встановити колір тексту та фону у вікні:

Color n n

де n – відповідна цифра (0 – чорний, 1 – синій, 2 – зелений, 3 – голубий, 4 – червоний, 5 – фіолетовий, 6 – жовтий, 7 – білий, 8 – сірий, 9 - синій).

Команда Cls видалить з консолі усе, що виводилося до цієї команди

Exit – закрити вікно.

команда1 & команда2 – послідовне виконання двох команд.

команда1 && команда2 – друга команда виконається тільки в разі успішного виконання першої команди

msg \* text – Pop-up вікно з текстовим повідомленням.

help команда – Вивід довідки з певної команди.

1.4.4 Паузи між командами

Іноді необхідно встановити певні проміжки часу між виконанням послідовності команд. Наприклад, можна задати час очікування підтвердження користувачем виконання наступного кроку, якщо користувач нічого не введе, то продовжити виконання операцій. Таку функцію можна реалізувати за допомогою команди:

ping -n 1 -w 500 192.168.254.254 >nul

-n 1 – скільки разів відправляється запит на IP адресу

-w 500 – час в мілісекундах

192.168.254.254 – довільна IP адреса

>nul – вивід результатів нікуди не здійснюється.

1.4.5 Системні налаштування

З командного рядка, а отже і з командного файлу, можна отримати основну інформацію про системні налаштування комп’ютера за допомогою наступного набору команд:

Date – Показати системну дату

Date 11.11.11 – Встановити дату

Time – Показати системний час

Time 00:00 – Встановити системний час

format диск /q - Швидке форматування диску

Mem – Відображає використану та доступну пам’ять в системі

DRIVERQUERY – Інформація про драйвери системи

Ver – Інформація про ОС

rundll32 keyboard,disable – Відключає клавіатуру(також можливі параметри mouse та enable)

Shutdown – Вимкнення комп’ютера.

1.4.6 Команда call

Команда call використовується для виклику з командного файлу іншого командного файлу з можливістю передачі йому вхідних параметрів:

call test.bat param1 param2.

Також, за умови увімкненого режиму розширеної обробки команд, за допомогою команди call можна використовувати мітки в якості адресату виклику. Синтаксис: call :мітка аргументи. Якщо в командному файлі ввімкнути режим розширеної обробки команд, це дає можливість використовувати додаткові ключі вбудованих команд.

Увімкнути режим розширеної обробки команд можна за допомогою команди

SetLocal EnableExtensions.

Команда SetLocal означає, що усі змінні, оголошені всередині командного файлу, будуть локальними, тобто їх значення після завершення його виконання будуть недоступними.

Під час виклику команди call за міткою створюється новий контекст поточного командного файлу з заданими аргументами. Під час використання виклику процедури за міткою в командному файлі, існують деякі особливості. Наприклад, створимо командний файл, в якому маємо підпрограму, яка обчислює суму двох чисел, переданих їй у якості параметрів, і задає отримане значення змінній res:

@echo off

call :SUM 100 22

echo The result is %res%

pause

exit

:SUM

set /a res=%1+%2

exit /b

В цьому прикладі використовується команда exit з параметром /b всередині процедури SUM. Якщо даний ключ не вказати, то замість виходу з процедури буде виконаний вихід із командного файлу (завершення CMD.exe). В команді set ми використали ключ /a, який означає, що стрічка праворуч від знаку рівності – чисельне висловлювання, яке треба обрахувати.

1.4.7 Перенаправлення вводу/виводу даних

Якщо необхідно зберігати результати виконання програми у текстовий файл, це можна зробити за допомогою спеціального символу >, або >>. Наприклад:

echo Hello>log.txt

echo Hello, World>>log.txt

Перша команда створює новій файл log.txt, якщо він не знайдений, і записує текст у файл. При цьому файл буде перезаписаний наново при кожному наступному виклику команди. Команда >> при багаторазовому виклику додає новий текст кожен раз на нову стрічку.

Якщо командний файл досить великий, зручно вести журнал з результатами виконаних команд. Тоді довелося б в кінці кожної команди додавати “>>%LOGFILE%”. Проте є спосіб значно спростити цю задачу. Для цього розглянемо приклад генерації порожньої html сторінки.

@echo off

set OUTPUT=out.html

if "%STDOUT\_REDIRECTED%" == "" (

set STDOUT\_REDIRECTED=yes

cmd.exe /c %0 %\* >%OUTPUT%

exit /b %ERRORLEVEL%

)

echo ^<html^>

echo ^<head^>

echo ^<title^>Test^</title^>

echo ^</head^>

echo ^<body^>

echo Hello World!

echo ^</body^>

echo ^</html^>

В результаті виконання такого командного файлу ми отримаємо файл out.html, в якому буде записано структуру простої html сторінки. В цьому прикладі також проілюстровано використання спеціального символу екранування “^”. Він використовується для того, щоб інтерпретатор розрізнював, наприклад символ відкриття тегу та перенаправлення виводу.

1.5 Ключі запуску

За допомогою команди start можна не тільки запускати програми, а й задавати їй додаткові параметри при запуску. Ключі вказуються через прямий слеш (/) після основної команди. Для різних програм існує багато наборів ключей, але є декілька загальних (наприклад, ключ довідки /? або /help). Наприклад, можна запустити програму у згорнутому вікні.

start /min winword

Важливим є ключ очікування завершення виконання команди перед подальшим виконанням командного файлу /wait. Наприклад, якщо потрібно розпакувати архів з файлом а потім цей файл запустити. В командному файлі буде записано дві команди – на розпакування та на запуск. Так як команди при виконанні командного файлу виконуються послідовно і без пауз, розпакування архіву може не встигнути завершитись, і файл не буде відкритий. В такому разі у першій команді треба вказати ключ /wait.

Часто командні файли використовуються для автоматизації процесу інсталяції програмного забезпечення. В такому разі корисним буде ключ, який запускає файл інсталяції програми в тихому режимі – тобто користувачу не потрібно вводити ніяких додаткових даних. В залежності від конкретного типу інсталятора, який використовується для встановлення програми на комп’ютер, Для цього можуть використовуватися різні ключі: /s, /q, /silent та інші.

1.6 Змінні

Командні файли Windows підтримують використання змінних. Для оголошення змінної та одночасного присвоєння їй значення застосовується команда set. Задати значення змінної можна відразу після її оголошення в командному файлі, або надати можливість користувачу самостійно надати значення змінної, використавши ключ /p. Для вилучення значення змінної її ім’я поміщують між двома знаками відсотка, як показано нижче:

set a=hello

set /p /a x=”Enter a number: ”

echo %a%

echo %x%

Під час роботи з командним файлом можна використовувати вбудовані змінні, такі як:

%RANDOM% - випадкове ціле число з проміжку [0, 32767]

%CD% - стрічка з шляхом поточної директорії

%DATE% - поточна системна дата

%TIME% - системний час

%SYSTEMDRIVE% - системний диск

%USERPROFILE% - шлях до домашньої директорії користувача

При використанні змінних оточення в командних файлах є повне обмеження, пов’язане з тим, що значення, яке було надане змінній залишається незмінним під час його модифікації всередині групи команд, що задається дужками, наприклад в командах if або for. Для уникнення даного обмеження використовується запуск командного процесора з параметром /v:on і замість знаків відсотка, для одержання значення змінної, використовуються символи знаку оклику !. Говорячи про командні файли, для отримання такої можливості слід увімкнути режим розширеної обробки команд командою

SetLocal EnableDelayedExpansion

Різницю в результаті використання значень змінних можна прослідкувати в наступному прикладі командного файлу:

@echo off

Setlocal EnableDelayedExpansion

set VAR=before

if "%VAR%" == "before" (

set VAR=after

echo using %% %% : =%VAR% , using ^! ^!: =!VAR!

)

pause

Команда set VAR=after виконується всередині підпрограми, обмеженої дужками і, якщо прибрати команду Setlocal EnableDelayedExpansion або не використовувати для отримання значення змінної VAR знаки оклику, її значення залишиться старим, тобто таким, яким воно було встановлено до входу в підпрограму. Аналогічна проблема спостерігається і тоді, коли значення змінної змінюється всередині циклу команди for.

Наприклад, для отримання списку файлів в поточному каталозі наступний командний файл буде працювати некоректно, адже значення змінної LIST всередині циклу буде незмінним.

set LIST=

for %%i in (\*) do set LIST=%LIST% %%i

@echo %LIST%

Для уникнення такої помилки слід змінити командний файл наступним чином:

Setlocal EnableDelayedExpansion

set LIST=

for %%i in (\*) do set LIST=!LIST! %%i

@echo %LIST%

Тепер значення змінної LIST всередині циклу for буде змінюватися, послідовно приймаючи значення імен файлів, розділених пробілом.

1.7 Конструкція goto

Команда goto перенаправляє виконання командного файлу на стрічку, позначену відповідною міткою. Синтаксис:

goto :мітка

Якщо в коді командного файлу не буде знайдено вказаної мітки, то виконання програми буде припинено, а на екран буде виведено повідомлення про помилку. Також слід зауважити, що команда goto розпізнає тільки перші вісім знаків мітки. Тобто, наступні мітки є еквівалентними і будуть розпізнані як перша мітка, що зустрічається в командному файлі:

:000000000

:000000001

:000000002

Також є вбудована мітка :eof, яка перенаправляє виконання в кінець командного файлу.

Команду goto можна використовувати разом з іншими командами для виконання операцій з умовами. Проте використання goto, наприклад, у циклі for призведе до неправильної роботи циклу, адже після виконання команд, зазначених після мітки, виконання циклу буде закінчене.

За допомогою команди goto можна зімітувати роботу циклу while:

:loop   
 if <умова> <goto exit>   
 <операції>   
 goto loop

:exit

…

1.8 Оператор умови if

if – оператор умовного виконання команд в командних файлах Windows. Команда з умовним оператором if має наступний вигляд:

if [not] <умова> команда

Як і в звичайних мовах програмування, підтримується ключове слово else, після якого вказуються команди, які будуть виконуватися в разі, якщо умова не виконується. Приклад:

if %var1%==%var2% (

echo ok

) else (

echo error

)

При увімкненому режимі розширеної обробки команд доступні наступні додаткові оператори порівняння:

**EQL** - еквівалентно

**NEQ** - не еквівалентно

**LSS** - менше

**LEQ** – менше або дорівнює

**GTR** - більше

**GEQ** – більше або дорівнює

1.9 Цикл for

Команда for використовується для виконання, вказаної в якості параметра, команди для кожного елементу з набору. Такими елементами можуть бути файли, каталоги, текстові стрічки. Синтаксис команди:

for %i in (набір) do команда [параметри]

%i – Будь-яка однобуквена змінна, під час виконання циклу їй присвоюється значення поточного елемента.

команда – Команда, яку треба виконати для кожного елементу з набору.

параметри – Параметри для команди.

Підтримуються також додаткові ключі для команди for:

/d – в якості набору задаються імена каталогів (не файлів)

/r – задає виконання команди для вказаного каталогу а також для всіх підкаталогів вказаного шляху.

for /L %i in (start, step, end) do команда [параметри]

Ключ /L задає обробку набору зі вказаними параметрами. start – початок, end – кінець, step – шаг. Наприклад, набір (6, -1, 1) розкриється в (6 5 4 3 2 1).

Якщо під час роботи з файлами всередині циклу for треба отримати певні відомості про поточний файл, можна використати наступні синтаксичні конструкції, в яких можна отримати певні дані зі змінної:

%~I - шлях до файлу без лапок

%~fI - повне ім'я файлу

%~dI - ім'я диски зі шляху до файлу

%~pI - шлях до файлу

%~nI - назва файлу

%~xI - формат файлу

%~aI - атрибути файла

%~tI - дата/час створення файлу

Для виконання декількох команд в тілі циклу, використовується наступна конструкція:

for %i in (набір) do (

команда1

команда2…

)

1.9 Обробка коду завершення програми

Код завершення виконання програми (errorlevel) використовується в командних файлах для моніторингу правильного виконання команд. Змінна errorlevel приймає числове значення в залежності від останньої виконаної команди. Зазвичай, якщо програма завершилася без помилок, значення дорівнює 0, якщо ж при програма завершила свою роботу з помилкою, змінна набуває значення, відмінне від 0 (1, 2, і т.д.) в залежності від типу помилки.

Для перевірки значення errorlevel найчастіше використовується команда if errorlevel n команда. Така перевірка повертає істинну, якщо код помилки n або більше, для відстеження конкретного значення потрібно починати перевірку з найбільшого очікуваного значення. Помилка, коли значення починають перевіряти найменшого – одна з найрозповсюдженіших при написанні командних файлів.

Приклади правильної обробки errorlevel:

1)

@echo off

rem запускаємо программу prg1.exe

prg1.exe

rem аналіз коду завершення

if errorlevel 2 goto filenotfound

if errorlevel 1 goto writeerror

if errorlevel 0 goto exitok

goto onexit

:filenotfound

echo Помилка! Файл не знайдено.

goto onexit

:writeerror

echo Помилка запису!

goto onexit

:exitok

echo Програма завершилась без помилок!

goto onexit

:onexit

pause

2)

@echo off

prg1.exe

if errorlevel 1 (

goto error

) else goto okay

goto onexit

:error echo error

goto onexit

:okay echo all ok

goto onexit

:onexit

pause

Також можна перевірити значення коду завершення, якщо використати змінну %errorlevel% (варто зазначити, що не треба створювати нову змінну, бо в такому разі це буде проста змінна оточення). Таким чином можна вивести код завершення на екран:

echo %errorlevel%.

**2. Windows Script Host**

2.1 Огляд WHS

Для написання повноцінних сценаріїв, працюючих під управлінням Windows, варто використовувати розроблений фірмою Microsoft сервер сценаріїв Windows (Windows Script Host, WHS).

Windows Script Host - компонент Microsoft Windows, призначений для запуску сценаріїв на скриптових мовах JScript і VBScript, а також і на інших додатково встановлюваних мовах (наприклад, Perl).

Можливості сценаріїв на WSH значно перевершують можливості командних файлів: є повноцінні мови з об'єктними можливостями, повний набір операцій з рядками, включаючи регулярні вирази, взаємодія з деякими програмами, що реалізують об'єктний скриптова інтерфейс (Active Scripting або OLE Automation), доступ до методів і властивостей їх об'єктів; операції з файлами і каталогами, обробка текстових файлів, маніпуляції з системним реєстром і т. п.

Сценарії з .js, .vbs, .wsf і деяких інших типів файлів запускаються за допомогою однієї з двох програм-активаторів: wscript.exe (з графічним інтерфейсом) або cscript.exe (з консольним інтерфейсом).

Найперші версії WSH поставлялися у вигляді окремого компонента, опціонально встановлюваного в Windows, але вже починаючи з Windows 98 WSH став частиною всіх ОС Windows.

2.2 Практичне використання

За допомогою WSH можна запускати сценарії, написані на будь-якій мові, для якої встановлено відповідний модуль (scripting engine), який підтримує технологію ActiveX Scripting.

Використовуючи сценарії WSH, можна працювати з файловою системою комп’ютера або керувати роботою інших програм (серверів автоматизації). При цьому можливості сценаріїв обмежені тільки засобами, які надаються серверами автоматизації.

Стандартними мовами, які використовуються для написання сценаріїв, є VBScript та JScript. Функціональність цих мов практично ідентична, але синтаксис дещо відрізняється: VBScript заснований на мові Visual Basic, а JScript схожий на Java та C.

**3. Застосування командних файлів Windows**

3.1 Для чого використовуються командні файли

Як вже зазначалося раніше, командні файли – це текстові файли, в яких записані допустимі команди операційної системи, а також деякі додаткові інструкції і ключові слова, що надають командним файлам певну схожість з алгоритмічними мовами програмування. Наприклад, якщо записати у файл clean.bat наступні команди:

cd %temp%

attrib –r \*.tmp

del \*.tmp

і запустити його на виконання, ти будуть видалені усі файли з розширенням tmp в тимчасовому каталозі Windows. Таким чином, виконання командного файлу призводить до того ж самого результату, що і послідовне виконання записаних у ньому команд. Отже, якщо потрібно часто виконувати одні й ті самі дії, то використання командних файлів може зекономити багато часу.

Починаючі з версій операційної системи Windows 9.x командні файли мали наступні можливості:

* виконання усіх команд операційної системи;
* виклик зовнішніх командних файлів;
* використання циклів для виконання однієї команди (в тому числі і для обробки набору файлів);
* використання оператора if;
* перехід на задану команду всередині поточного файлу.

В командних файлах Windows NT з’явилося доволі багато нових можливостей, серед них:

* вдосконалена робота умовного оператора if (поява операторів умови, відмінних від порівняння на рівність);
* механізм переходу до заданої мітки в новій копії поточного командного файлу (з довільними параметрами командного рядка);
* команди з пакету Windows NT Resource Kit дозволяють значно розширюють функціонал командних файлів.

3.2 Утиліти Windows NT Resource Kit

Windows NT Resource Kit – це пакет утиліт, розроблений для операційних систем Windows NT. В контексті роботи з командними файлами варто звернути увагу на наступні утиліти: forfiles, choice, logtime, now, timethis, sleep, timeout, freedisk.

Команда forfiles є зручним засобом виконання заданих команд для групи файлів в командних файлах. Формат запису:

forfiles [/p шлях] [/m маска] [/s] [/c команда] [/d [+/-] {dd.mm.yyy | dd}]

Параметр /d визначає вибір файлів, у яких дата останньої правки більша абр менша від вказаної, або файлів, у яких дата останньої правки відрязняється від поточної системної на задану кількість днів (в межах від 0 до 32768), “+” використовуються за замовченням, якщо знак не вказано.

За допомогою команди choice можна обробити ввід користувача, наприклад:

1)

choice

goto Answer%errorlevel%

:Answer1

echo case 1

:Answer2

echo case 2

2)

choise /c:ync Yes, No or Cancel

Ключ /c: задає можливі варіанти відповіді, після цього вказується текст, який буде виведено на екран. В результаті виконання другого прикладу на екран буде виведено наступний запит: Yes, No or Cancel [Y,N,C]?

Якщо задати ключ /t[:]c,nn, то команда choice перед використанням заданої за замовченням варіанту c зробить паузу впродовж заданого числа секунд (в проміжку від 0 до 99, 0 – необмежена пауза).

Команда logtime дозволяє відстежувати час, який витрачається на виконання певних команд. В результаті своєї роботи команда logtime створює файл-протокол з назвою logtime.log, в який записуються поточні дата і час, а також заданий текстовий коментар. Якщо файл logtime.log вже існує, то нова інформація буде додана в кінець файлу. Приклад:

logtime “початок виконання”

команди

logtime “кінець виконання”

Утиліта now схожа на команду echo, але з додатковою інформацією про поточну дату і час. На відміну від команди logtime, результат виконання команди now можна перенаправити у заданий файл:

now “початок виконання” >> log.txt

команди

now “кінець виконання” >> log.txt

Ще однією командою для відстеження часу виконання програм, є timethis. Ця утиліта дозволяє встановити, скільки часу потребує система на виконання тої чи іншої команди або програми. Синтаксис:

timethis команда

Таким чином, буде виконана сама команда, і виведено час початку і завершення виконання, а також витрачений час, з точністю до тисячної долі секунди. Інформацію, отриману командою, можна записати в файл. Якщо ж символи перенаправлення мають входити в саму команду, то таку команду треба взяти у лапки, наприклад:

timethis “dir > dirlist.txt”

Наступні дві утиліти, sleep і timeout, дозволяють встановити паузу у виконанні командного файлу на певний час. Синтаксис команди sleep:

sleep n

де n – тривалість паузи в секундах. Команда timeout є схожою на команду sleep, і має такий самий синтаксис, проте користувач має змогу припинити паузу, натиснувши будь-яку клавішу.

Утиліта freedisk перевіряє наявність вільного місця на диску. Синтаксис команди:

freedisk диск об’єм

Параметр диск задає ім’я диску, а об’єм виконує запит на вказану (в байтах) кількість вільної пам’яті.

3.3 Приклади застосування командних файлів

3.3.1 Утиліта для пошуку файлів

Наступна утиліта виконує пошук файлу за заданим ім’ям вглиб вказаного каталогу і виводить результат у текстовий файл.

@echo off

@title Finder

color 07

:A

echo enter the root directory to search for:

set /p rdir=

echo enter the name of file to search:

set /p name=

dir /b /s /a-d %rdir%:\%name%.\* > fbN\_Output.txt

echo ====================DONE====================

goto A

pause

Якщо замінити маску імені файлів на \*%name%\*.\* то будуть знайдені усі файли, в назві яких присутня задана стрічка.

3.3.2 Утиліта для пошуку папок

Цей приклад командного файлу виконує пошук папки за частиною її назви по всім дискам.

@echo off

@title Finder

SetLocal EnableExtensions EnableDelayedExpansion

color 07

Set /p "Seek=enter the part of name of the directory to search for: "

For /F "tokens=2 delims==" %%D in ('wmic logicaldisk where drivetype^=3 get caption /value^| find /i "Caption"') do (

set Disk=%%D

set Disk=!Disk:~0,2!

set Again=

for /f "delims=" %%? in ('dir /B /AD "!Disk!\\*" ^| find /i "%Seek%"') do (

if not Defined Again (

set Again=true

echo.

echo Drive !Disk!

echo ============

)

Set /A C+=1

Set Folder.!C!=!Disk!\%%?

echo !C!. %%?

))

set /p "ch=Select the number to open it: "

start "" "!Folder.%ch%!"

3.3.3 Утиліта для пошуку файлів за форматом

Пошук виконується у всіх підкаталогах вказаного шляху, у текстовий файл записуються шлях до всіх знайдених файлів зі вказаним розширенням.

@echo off

@title Finder

color 07

:A

echo enter the root directory to search for:

set /p rdir=

echo enter the extension of files to search:

set /p ext=

rem /s=відображення файлів з каталогу і всіх підкаталогів /a-d=не перевіряти каталоги

dir /b /s /a-d %rdir%:\\*.%ext% > fbE\_Output.txt

echo ====================DONE====================

goto A

pause

3.3.4 Створення нового файлу

Наступний приклад ілюструє роботу з параметрами виклику командних файлів. Для створення нового порожнього файлу треба зберегти файл у директорію, шлях до якої вказаний у змінній PATH (наприклад, C:\Windows\System32\, ім'я файлу – nf.bat). Тоді для виклику командного файлу достатньо буде ввести команду nf f:\newFile.avi.

@echo off

:: Create an empty file whose name is given in the command line

if "%1" EQU "" goto error

copy nul %1

goto exit

:error

ECHO ERROR: You must give some name for a new file!

:exit

3.3.5 Автоматизований вхід на електронну пошту

Ця утиліта дозволяє швидко відкрити сторінку з авторизацією в електронній пошті (наприклад Gmail), після цього у буфер обміну копіюється вказаний в командному файлі логін користувача, на екран виводиться відповідне повідомлення. Через вказаний проміжок часу (20 с.), в буфер копіюється пароль від облікового запису, потім буфер обміну очищується.

@Echo off

@title Email login

color 07

::The page with your mail box authorization

start https://accounts.google.com/ServiceLogin

ping -n 20 127.0.0.1 > nul

::Put your login here

msg \* Enter Login

<nul set /p x="w1nner1715"| Clip

ping -n 20 127.0.0.1 > nul

::Put your password here

msg \* Enter Password

<nul set /p x="120796nastia"| Clip

ping -n 20 127.0.0.1 > nul

msg \* The clipboard is cleared

<nul set /p x=""| Clip

exit

3.3.6 Менеджер задач

Ця утиліта виводить на екран список усіх запущених в системі процесів, та дозволяє припинити виконання будь-якого з них, вказавши його номер.

@echo off

@title Task Killer

SetLocal EnableDelayedExpansion

For /F "tokens=1 delims=," %%p In ('tasklist /FO CSV^|sort') Do (

Set nam=%%p

if %%p neq "" (

Set /A nn +=1

set proc!nn!=%%p

if !nn! lss 10 (Echo ^ !nn!. !nam:~1,-1!) else (Echo !nn!. !nam:~1,-1!)

))

:start

set /p kill=Enter the number of task you want to stop:

if "%kill%"=="" goto :start

echo You have chosen !proc%kill%!. Are you shure?

echo.

pause

taskkill /f /im !proc%kill%! /T

pause

3.3.7 Видалення тимчасових файлів

@echo off

@title Cleaner

set /p con="Continue y/n %?"

if %con%==y (

goto goon

) else goto end

:goon

DEL /F /S /Q %WINDIR%\Temp\\*.\*

DEL /F /S /Q %SYSTEMDRIVE%\Temp\\*.\*

DEL /F /S /Q %Temp%\\*.\*

DEL /F /S /Q %Tmp%\\*.\*

DEL /F /S /Q %WINDIR%\Prefetch\\*.\*

DEL /F /S /Q %SYSTEMDRIVE%\\*.log

DEL /F /S /Q %SYSTEMDRIVE%\\*.bak

DEL /F /S /Q %SYSTEMDRIVE%\\*.gid

:end

pause

3.3.8 Автоматичне створення резервних копій

Ця утиліта виявиться корисною під час роботи з важливими документами. У командному файлі вказується шлях до папки, резервну копію якої необхідно створити. Також задається місце зберігання самих архівів. В результаті виконання буде створено папку (ім’я – поточна дата), в якій буде міститися архів відповідної директорії (назва – поточний час).

chcp 1251

@echo off

@title AutoBackups

color 07

set FROM=C:\Users\Serhii\Documents

set TO=C:\Users\Serhii\backups

rem Створення каталогу для зберігання

md "%TO%\"

rem Формування імя каталогуз поточної дати

set VDATE=%date:~-10%

rem Ім'я архіву - поточна дата - 12:00:00.99

rem відкинути соті долі секунди, заміна : на . - 12.00.00

set vtime=%TIME:~0,-3%

set vtime=%vtime::=.%

rem Створення підкаталогу, ім'я - поточна дата

md "%TO%\%VDATE%"

:: команда для архівування, ключ -r - для арх. з вкладеними папками

:: -mn, n - 1-5 степінь стискання

rar.exe a -r -m3 "%TO%\%VDATE%\%VTIME%.rar" "%FROM%\\*.\*"

pause

3.3.9 Перевірка відповідей учнівських робіт

Наступна утиліта призначена для автоматизованої перевірки великої кількості текстових файлів. Задається файл, що приймається як правильний варіант відповіді. Усі текстові файли в заданому каталозі порівнюються із правильним. Якщо вони ідентичні, то у вихідному файлі робиться відповідний запис, якщо наявні відмінності, то робота вважається виконаною неправильно.

@echo off

@title Comparator

SetLocal EnableExtensions EnableDelayedExpansion

color 07

set Mask=\*.txt

:: to check

set tc="answers\\*.txt"

:: right answers

set ra="rightAnswers.txt"

FOR %%f IN (%tc%) DO (

fc "%%f" "%ra%"

IF ERRORLEVEL 1 (

echo %%~nf is wrong>>results.txt

) ELSE echo %%~nf is right>>results.txt

)

pause

**Висновки**

Під час виконання цієї роботи я поліпшив свої знання у створенні та особливостях використання командних файлів Windows. Я поглибив свої знання стосовно роботи з командним рядком і застосував теоретичні знання на багатьох практичних прикладах. Під час виконання роботи було розроблено ряд командних файлів для практичного пояснення до наявного теоретичного матеріалу, а також декілька сервісних утиліт, які будуть корисними при роботі з операційною системою Windows.

**Список використаної літератури**

1. Попов А. Командные файлы и сценарии Windows Script Host. / Попов А.; СПб - «БХВ-Петербург», 2002. – 320с.

2. Уильям Р. Станек. Командная строка Microsoft Windows. / Уильям Р.; СПб - «БХВ-Петербург», 2004. – 480с.

3. RSDN Magazine. “Урок bat-аники” [Електронний ресурс].

URL: <http://rsdn.ru/article/winshell/batanyca.xml#ERBAC> (дата звернення: 24.04.2015)

4. “Писать bat-файлы может каждый!” [Електронний ресурс].

URL: <http://www.philosoft.ru/batniki.zhtml> (дата звернення: 24.04.2015)

5. Командные файлы Windows (Справочник) [Електронний ресурс].

URL: <http://ab57.ru/cmd.html> (дата звернення: 24.04.2015)

6. Wikipedia the free encyclopedia [Електронний ресурс].

URL: <https://en.wikipedia.org/wiki/Batch_file> (дата звернення: 24.04.2015)

URL: <https://en.wikipedia.org/wiki/Windows_Script_Host> (дата звернення: 09.05.2015)

7. Общие сведения о командной оболочке [Електронний ресурс].

URL: <http://cmdhelp.ru> (дата звернення: 24.04.2015)

8. [CMD — командная строка как 2×2](http://cmd-command.ru/) [Електронний ресурс].

URL: <http://cmd-command.ru/> (дата звернення: 09.05.2015)

9. Командная строка Windows[Електронний ресурс].

URL: <http://www.windowsfaq.ru/content/category/3/19/57> (дата звернення: 09.05.2015)

10. Памятка по развитым командным файлам MS-DOS (bat-файлам) [Електронний ресурс].

URL: <http://www.philosoft.ru/batch-files-programming-memo.zhtml> (дата звернення: 09.05.2015)